UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/802,586	03/17/2004	Joel David Munter	MP1502	5149

64768 7590 11/24/2009

MARSHALL, GERSTEIN & BORUN, LLP (MARVELL)
233 SOUTH WACKER DRIVE
6300 SEARS TOWER
CHICAGO, IL 60606-6357

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
11/24/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/802,586	MUNTER ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 July 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-4,7-11,13-18,20-26 and 28-33* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-4, 7-11, 13-18, 20-26, and 28-33* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated July 13, 2009, responding to the Office Action mailed April 15, 2009 provided in the rejection of claims 1-11, 13-18, 20-26, and 28-33, wherein claims 1, 8, 11, 15, 18, 22-24, 26, and 31-33 have been amended; and claims 5 and 6 have been canceled.

Claims 1-4, 7-11, 13-18, 20-26, and 28-33 remain pending in the application and which have been fully considered by the examiner.

The status of claim 1 and 15 were inadvertently listed as "Previously Presented" instead of –Currently Amended--

Applicant's arguments with respect to claims currently amended have been fully considered but are moot in view of the new grounds of rejection – see *Chauvel* et al. - art previously applied, as applied hereto.

With previous amendments, the claim limitations were focusing on generating object code, for example, Claim 1 is generally directed to a method that includes "compiling plurality of non-native instructions to generate object code for the non-native instructions, wherein compiling the plurality of non-native instructions includes replacing an object code segment from the generated object code with an alternative object code segment …" (emphasis added - stated on page 11 of REMARKS dated December 18, 2008); and now Applicant added more specified profiling mechanism to the independent claims. Further, *Chauvel* et al. (*Application Execution Profiling in Conjunction with a Virtual Machine*) indeed teaches a need for a method and apparatus for reliably estimating

performance characteristics, such as execution time and energy, in a device using a virtual machine interface (e.g., [0011] – emphasis added). Therefore, *Chauvel* et al. is re-applied to this Office Action accordingly.

2. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a).

Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the

art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-4, 7-11, 13-18, 20-26, and 28-33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Chauvel et al. (Pub. No. US 2004/0010785 A1) (hereinafter 'Chauvel' – art of record) in view of Chheda et al. (Pub. No. US 2005/0114850 A1) (hereinafter 'Chheda')

4. **As to claim 1** (Currently Amended), Chauvel discloses a method comprising:

- receiving a plurality of non-native instructions in a selected one of a source form (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added);
- determining an initial number of times to interpretively execute the plurality of non-native instructions (e.g., [0012] - … by acquiring an application profile that specifies a number of executions of a plurality of operations used in the specified portion of the application … - emphasis added);
- interpretively executing the plurality of non-native instructions the initial number of times (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the

application 12, or in specified parts of the application 12 ... – emphasis added); and

- monitoring execution of the plurality of non-native instructions to determine when the plurality of non-native instructions have been interpretively executed the initial number of times (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior);

Further, Chauvel discloses a profiling system independently creates application profiles that indicate the number of executions of each operation in the application and virtual machine profile which indicate the time/energy consumed by each operation on a particular hardware platform (e.g., Abstract – emphasis added) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Focused Re-compilation of Executables and Hardware Mechanisms based on Compiler-Architecture Interaction and Compiler-Inserted Control*, Chheda discloses:

- compiling the plurality of non-native instructions to generate object code for the non-native instructions only after interpretively executing the plurality of non-native instructions the initial number of times, wherein compiling the plurality of non-native instructions includes replacing an object code segment from the generated object code with an alternative object code segment if the alternative object code

segment improves at least a selected one of an execution power level required and an execution energy level required to execute the generated object code in a target execution environment (e.g., [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chheda into the Chauvel's system to further provide other limitations stated above in the Chauvel system.

The motivation is that it would further enhance the Chauvel's system by taking, advancing and/or incorporating the Chheda's system which offers significant advantages that due to the compiler-driven nature of the solution, the impact of control overhead can be kept very small; such control energy overhead could kept below 1%-2%, if instruction memory energy optimization are included, while providing energy optimizations in the range of 30%-68% if several techniques in different processor domains/component are included as once suggested by Chheda (e.g., [0031] – emphasis added)

5. **As to claim 2** (Original) (incorporating the rejection in claim 1), Chauvel discloses the method wherein said receiving comprises receiving the non-native instructions in a byte code form (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added)

6. **As to claim 3** (Previously Presented) (incorporating the rejection in claim 1), Chheda discloses the method wherein said compiling comprises analyzing the object code segment for execution power level requirement, and determining whether an alternative object code segment with lower execution power level requirement is available (e.g., Fig. 3; [0075] – Disambiguating an Executable; [0076] - … the executable file is analyzed by a binary scanning tool in order to gain information about the various section and any other symbolic information that may be available. This information is used in order to create a version of the program based on energy-focused Binary Intermediate Format or BIF …; [0077]; [0078] – Once program analyses and optimizations have been run, the optimized BIF object can be converted back into an executable of the same type as the original one or possibly another type with a different instruction – see 42 in Fig. 3)

7. **As to claim 4** (Previously Presented) (incorporating the rejection in claim 1), Chheda discloses the method wherein said compiling comprises analyzing the object code segment for execution energy consumption, and determining whether an alternative object code segment with lower execution energy

consumption is available (e.g., Fig. 3; [0075] – Disambiguating an Executable; [0076] - … the executable file is analyzed by a binary scanning tool in order to gain information about the various section and any other symbolic information that may be available. This information is used in order to create a version of the program based on energy-focused Binary Intermediate Format or BIF …; [0077]; [0078] – Once program analyses and optimizations have been run, the optimized BIF object can be converted back into an executable of the same type as the original one or possibly another type with a different instruction – see 42 in Fig. 3)

8. **As to claim 7** (Currently Amended) (incorporating the rejection in claim 6), Chauvel discloses the method wherein the method further comprises

- monitoring said compiling for power level required to perform compilation (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior);
- updating a current understanding of power level required for compilation (e.g., Fig. 3b – element 56 – For each Operation Divide Energy Consumption by Number of Executions and Store in JVM Profile; element 54 – Increment Associated Register with Energy Consumption; Update Maximum and Minimum Energy Values; [0013] - … an accurate estimation

of a performance criteria, such as average time, maximum time, or energy consumption, for the application can be provided …); and

- updating the initial number of times the plurality of non-native instructions are to be interpretively executed before compiling the received non-native instructions, if said monitoring observes a power level required for compilation to be different from the current understanding (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior; [0010] – To optimize an application, estimations of execution time or energy consumption are often needed …)

9. **As to claim 8** (Previously Presented) (incorporating the rejection in claim 6), Chauvel discloses the method wherein the method further comprises

- monitoring said compiling for amount of energy required to perform an average compilation (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior);
- updating a current understanding of amount of energy required for an average compilation (e.g., Fig. 3b – element 56 – For each Operation

Divide Energy Consumption by Number of Executions and Store in JVM Profile; element 54 – Increment Associated Register with Energy Consumption; Update Maximum and Minimum Energy Values; [0013] - … an accurate estimation of a performance criteria, such as average time, maximum time, or energy consumption, for the application can be provided …); and

- updating the initial number of times the plurality of non-native instructions are to be interpretively executed before compiling the received non-native instructions, if said monitoring observes an amount of energy required for compilation to be different from the current understanding (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior; [0010] – To optimize an application, estimations of execution time or energy consumption are often needed …)

10. **As to claim 9** (Original) (incorporating the rejection in claim 1), Chauvel discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for power level required to execute the native instructions (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and

Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior); and

- updating power level requirements of selected ones of the native instructions if said monitoring observes power level requirements for the selected ones of the native instructions to be different from current understandings of the power level requirements of the selected ones of the native instructions (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior; [0010] – To optimize an application, estimations of execution time or energy consumption are often needed …)

11. **As to claim 10** (Original) (incorporating the rejection in claim 1), Chauvel discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for amount of energy required to execute the native instructions (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes

that have a wide variant must be separately monitored for wide and normal behavior); and

- updating energy requirements of selected ones of the native instructions if said monitoring observes energy requirements for the selected ones of the native instructions to be different from current understandings of the energy requirements of the selected ones of the native instructions (e.g., Fig. 3b – element 56 – For each Operation Divide Energy Consumption by Number of Executions and Store in JVM Profile; element 54 – Increment Associated Register with Energy Consumption; Update Maximum and Minimum Energy Values; [0013] - … an accurate estimation of a performance criteria, such as average time, maximum time, or energy consumption, for the application can be provided …)

12. **As to claim 11** (Currently Amended), Chauvel discloses in an electronic device, a method of operation, comprising:

- receiving a plurality of non-native instructions (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added);
- determining an initial number of times to interpretively execute the non-native instructions based at least in part on one or more of an expected power level required to perform a compile or an expected energy required

to perform the compile (e.g., [0012] - … by acquiring an application profile that specifies a number of executions of a plurality of operations used in the specified portion of the application … - emphasis added);

- executing the non-native instructions for the initial number of times using an interpreter (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added); and
- monitoring execution of the non-native instructions to determine when the non-native instructions have been executed the initial number of times using the interpreter (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior);

Further, Chauvel discloses a profiling system independently creates application profiles that indicate the number of executions of each operation in the application and virtual machine profile which indicate the time/energy consumed by each operation on a particular hardware platform (e.g., Abstract – emphasis added) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Focused Re-compilation of Executables and Hardware Mechanisms based on Compiler-Architecture Interaction and Compiler-Inserted Control*, Chheda discloses:

- compiling the non-native instructions into object code only after executing the received non-native instructions for said initial number of times using the interpreter (e.g., [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chheda into the Chauvel's system to further provide other limitations stated above in the Chauvel system.

The motivation is that it would further enhance the Chauvel's system by taking, advancing and/or incorporating the Chheda's system which offers significant advantages that due to the compiler-driven nature of the solution, the impact of control overhead can be kept very small; such control energy overhead could kept below 1%-2%, if instruction memory energy optimization are included, while providing energy optimizations in the range of 30%-68% if several

techniques in different processor domains/component are included as once suggested by Chheda (e.g., [0031] – emphasis added)

13. **As to claim 13** (Previously Presented) (incorporating the rejection in claim 11), Chauvel discloses the method wherein the method further comprises

- monitoring said compiling for a compilation requirement employed in determining the initial number of times the received non-native instructions are to be executed using the interpreter before compiling (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior); and
- updating a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (e.g., Fig. 3b – element 56 – For each Operation Divide Energy Consumption by Number of Executions and Store in JVM Profile; element 54 – Increment Associated Register with Energy Consumption; Update Maximum and Minimum Energy Values; [0013] - … an accurate estimation of a performance criteria, such as average time, maximum time, or energy consumption, for the application can be provided …)

14. **As to claim 14** (Original) (incorporating the rejection in claim 11), Chauvel discloses the method wherein the generated object code comprises a plurality of native instructions, and the method further comprises

- monitoring execution of the generated object code for execution requirements of the native instructions (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior); and
- updating execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Fig. 3b – element 56 – For each Operation Divide Energy Consumption by Number of Executions and Store in JVM Profile; element 54 – Increment Associated Register with Energy Consumption; Update Maximum and Minimum Energy Values; [0013] - … an accurate estimation of a performance criteria, such as average time, maximum time, or energy consumption, for the application can be provided …)

15. **As to claim 15** (Currently Amended), Chauvel discloses an article of manufacture comprising:

- a computer readable medium; and
- a plurality of instructions designed to implement a compiler to:
    - determine an initial number of times to interpretively execute a plurality of non-native instructions (e.g., [0012] - … by acquiring an application profile that specifies a number of executions of a plurality of operations used in the specified portion of the application … - emphasis added);
    - interpretively execute the plurality non-native instructions the initial number of times (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added); and
- monitor execution of non-native instructions to determine when the non-native instructions have been interpretively executed the initial number of times (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior);

Further, Chauvel discloses a profiling system independently creates application profiles that indicate the number of executions of each operation in the application and virtual machine profile which indicate the time/energy

consumed by each operation on a particular hardware platform (e.g., Abstract – emphasis added) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Focused Re-compilation of Executables and Hardware Mechanisms based on Compiler-Architecture Interaction and Compiler-Inserted Control*, Chheda discloses:

- compile the non-native instructions to generate object code for the non-native instructions only after interpretively executing the plurality of non-native instructions the initial number of times, and
- replace a segment of the generated object code with an alternative object code segment if the alternative object code segment improves at least a selected one of an execution power level required and an execution energy required to execute the generated object code in a target execution environment (e.g., Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …; [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures)

16. **As to claim 16** (Currently Amended) (incorporating the rejection in claim 15), please refer to claim **3** as set forth accordingly.

17. **As to claim 17** (Currently Amended) (incorporating the rejection in claim 15), please refer to claim **4** as set forth accordingly.

18. **As to claim 18** (Currently Amended), Chauvel discloses an article of manufacture comprising:

a computer readable medium; and

a plurality of instructions designed to implement a runtime manager equipped to:

- receive a plurality of non-native instructions (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added),
- determine an initial number of times to interpretively execute the non-native instructions, the initial number of times based at least in part on one or more of an expected power level required to perform an average compile or an expected energy required to perform the average compile (e.g., [0012] - … by acquiring an application profile that specifies a number of executions of a plurality of operations used in the specified portion of the application … - emphasis added)

- execute the non-native instructions for said initial number of times using an interpreter (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added),

- monitor execution of the non-native instructions to determine when the non-native instructions have been interpretively executed the initial number of times (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior);

Further, Chauvel discloses a profiling system independently creates application profiles that indicate the number of executions of each operation in the application and virtual machine profile which indicate the time/energy consumed by each operation on a particular hardware platform (e.g., Abstract – emphasis added) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Focused Re-compilation of Executables and Hardware Mechanisms based on Compiler-Architecture Interaction and Compiler-Inserted Control*, Chheda discloses:

- invoke a compiler to compile the non-native instructions into object code after executing the received non-native instructions for said initial

number of times using the interpreter (e.g., Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …; [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chheda into the Chauvel's system to further provide other limitations stated above in the Chauvel system.

The motivation is that it would further enhance the Chauvel's system by taking, advancing and/or incorporating the Chheda's system which offers significant advantages that due to the compiler-driven nature of the solution, the impact of control overhead can be kept very small; such control energy overhead could kept below 1%-2%, if instruction memory energy optimization are included, while providing energy optimizations in the range of 30%-68% if several techniques in different processor domains/component are included as once suggested by Chheda (e.g., [0031] – emphasis added)

19. **As to claim 20** (Original) (incorporating the rejection in claim 18), Chheda discloses the article wherein the runtime manager is further equipped to

- monitor said compiling for a compilation requirement employed in determining the initial number of times received non-native instructions are to be executed before compiling; and
- update a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (e.g., Sec. 1 – Introduction, 5th Para - … The bytecodes are executed by Java Virtual Machine (JVM). Simple JVMs execute Java applications by interpreting their bytecodes …; Sec. 3 – Analysis of Execution and Compilation Strategies, 2nd Para - … be identified by automatic tools that make use of profile information …; Sec. 2 – Target Platforms and Benchmarks, 3rd Para - … tracks the energy consumptions in the process core (data-path), on-chip caches …; Sec. 3.1 Analysis of Static Strategy; Fig. 6 – Energy consumption of three benchmarks with static execution strategies)

20. **As to claim 21** (Original) (incorporating the rejection in claim 18), Chauvel discloses the article wherein the generated object code comprises a plurality of native instructions, and the runtime manager is further equipped to

- monitor execution of the generated object code for execution requirements of the native instructions (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM

Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior); and

- update execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Fig. 3b – element 56 – For each Operation Divide Energy Consumption by Number of Executions and Store in JVM Profile; element 54 – Increment Associated Register with Energy Consumption; Update Maximum and Minimum Energy Values; [0013] - … an accurate estimation of a performance criteria, such as average time, maximum time, or energy consumption, for the application can be provided …)

21. **As to claim 22** (Previously Presented), Chauvel discloses a system, comprising:

a storage medium having stored therein a plurality of instructions implementing a compiler [0063] to:

- receive a plurality of non-native instructions in a selected one of a source form and an intermediate form (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during

execution of the application 12, or in specified parts of the application 12 ... – emphasis added),

- determine an initial number of times to interpretively execute the plurality of non-native instructions (e.g., [0012] - … by acquiring an application profile that specifies a number of executions of a plurality of operations used in the specified portion of the application … - emphasis added);
- interpretively execute the plurality of non-native instructions the initial number of times (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added); and
- monitor execution of the non-native instructions to determine when the non-native instructions have been interpretively executed the initial number of times (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior);

Further, Chauvel discloses a profiling system independently creates application profiles that indicate the number of executions of each operation in the application and virtual machine profile which indicate the time/energy

consumed by each operation on a particular hardware platform (e.g., Abstract – emphasis added) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Focused Re-compilation of Executables and Hardware Mechanisms based on Compiler-Architecture Interaction and Compiler-Inserted Control*, Chheda discloses:

- compile the non-native instructions to generate object code for the non-native instructions only after interpretively executing the plurality of non-native instructions the initial number times (e.g., NOTE: Chauvel - Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added), and
- replace a segment of the generated object code with an alternative object code segment if the alternative object code segment improves at least a selected one of an execution power level required and an execution energy required to execute the generated object code in a target execution environment (e.g., [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy

efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures); and

- a processor coupled to the storage medium to execute the instructions implementing the compiler (e.g., [0020])

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chheda into the Chauvel's system to further provide other limitations stated above in the Chauvel system.

The motivation is that it would further enhance the Chauvel's system by taking, advancing and/or incorporating the Chheda's system which offers significant advantages that due to the compiler-driven nature of the solution, the impact of control overhead can be kept very small; such control energy overhead could kept below 1%-2%, if instruction memory energy optimization are included, while providing energy optimizations in the range of 30%-68% if several techniques in different processor domains/component are included as once suggested by Chheda (e.g., [0031] – emphasis added)

22. **As to claim 23** (Currently Amended) (incorporating the rejection in claim 22), Chheda discloses the system wherein said compiler analyzes the object code segment for execution power level requirement, and determines whether an alternative object code segment with lower execution power level requirement is available (e.g., Fig. 3; [0075] – Disambiguating an Executable; [0076] - … the executable file is analyzed by a binary scanning tool in order to gain information

about the various section and any other symbolic information that may be available. This information is used in order to create a version of the program based on energy-focused Binary Intermediate Format or BIF …; [0077]; [0078] – Once program analyses and optimizations have been run, the optimized BIF object can be converted back into an executable of the same type as the original one or possibly another type with a different instruction – see 42 in Fig. 3)

23. **As to claim 24** (Currently Amended) (incorporating the rejection in claim 22), Chheda discloses the system wherein said compiler analyzes the object code segment for execution energy consumption, and determines whether an alternative object code segment with lower execution energy consumption is available (e.g., Fig. 3; [0075] – Disambiguating an Executable; [0076] - … the executable file is analyzed by a binary scanning tool in order to gain information about the various section and any other symbolic information that may be available. This information is used in order to create a version of the program based on energy-focused Binary Intermediate Format or BIF …; [0077]; [0078] – Once program analyses and optimizations have been run, the optimized BIF object can be converted back into an executable of the same type as the original one or possibly another type with a different instruction – see 42 in Fig. 3)

24. **As to claim 25** (Original) (incorporating the rejection in claim 22), Chauvel discloses the system wherein the apparatus further comprises a wireless communication interface to receive the non-native instructions (e.g., [0010] - … This is particularly true in the case of mobile devices, such as smart phones,

personal digital assistants, and the like, which have limited energy and processing resources …)

25. **As to claim 26** (Currently Amended), Chauvel discloses a system, comprising:

a communication interface to receive a plurality of non-native instructions;

a storage medium coupled to the communication interface, and having stored therein a plurality of instructions designed to implement a runtime manager equipped to determine an initial number of times to:

- determine an initial number of times to interpretively execute the non-native instructions, the initial number of times based at least in part on one or more of an expected power level required to perform an average compile or an expected energy required to perm the average compile (e.g., [0012] - … by acquiring an application profile that specifies a number of executions of a plurality of operations used in the specified portion of the application … - emphasis added),
- execute the received non-native instructions for the initial number of times using an interpreter (e.g., Fig. 1a; [0024] - … The application profile 10 is a byte-code based profile which indicates how many times each operation (such as byte-code or a native method) is used during execution of the application 12, or in specified parts of the application 12 ... – emphasis added); and

- monitor execution of the non-native instructions to determine when the non-native instructions have been interpretively executed the initial number of times (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior);

Further, Chauvel discloses a profiling system independently creates application profiles that indicate the number of executions of each operation in the application and virtual machine profile which indicate the time/energy consumed by each operation on a particular hardware platform (e.g., Abstract – emphasis added) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Energy-Focused Re-compilation of Executables and Hardware Mechanisms based on Compiler-Architecture Interaction and Compiler-Inserted Control*, Chheda discloses:

- invoke a compiler to compile the non-native instructions into object code only after executing the received non-native instructions for said initial number of times using the interpreter (e.g., Fig. 2, element 24 – Executable Re-Compiler; [0088] – … the executable re-compiler 30 can be integrated with various source-level compilers in the front-end 48 that have source files 46 as inputs …; [0045] - … Many of these micro-operations are not necessary if there is related information extracted in the compiler that provides it, or if there is program information that enables a

way to replace these micro-operations with more energy-efficient but equivalent ones; [0050] - … inserting a more energy efficient instruction replacing an existing instruction and may include inserting control bits to control hardware structures); and

- a processor coupled to the storage medium to execute the instructions implementing the runtime manager (e.g., [0020])

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Chheda into the Chauvel's system to further provide other limitations stated above in the Chauvel system.

The motivation is that it would further enhance the Chauvel's system by taking, advancing and/or incorporating the Chheda's system which offers significant advantages that due to the compiler-driven nature of the solution, the impact of control overhead can be kept very small; such control energy overhead could kept below 1%-2%, if instruction memory energy optimization are included, while providing energy optimizations in the range of 30%-68% if several techniques in different processor domains/component are included as once suggested by Chheda (e.g., [0031] – emphasis added)

26. **As to claim 28** (Previously Presented) (incorporating the rejection in claim 26), Chauvel discloses the system wherein the runtime manager is further equipped to

- monitor said compiling for a compilation requirement employed in determining the initial number of times received non-native instructions are to be executed using the interpreter before compiling (e.g., Fig. 3b – element 56 – For Each Operation, Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior); and
- update a current understanding of the compilation requirement if said monitoring observes the compilation requirement to be different from the current understanding (e.g., Fig. 3b – element 56 – For each Operation Divide Energy Consumption by Number of Executions and Store in JVM Profile; element 54 – Increment Associated Register with Energy Consumption; Update Maximum and Minimum Energy Values; [0013] - … an accurate estimation of a performance criteria, such as average time, maximum time, or energy consumption, for the application can be provided …)

27. **As to claim 29** (Original) (incorporating the rejection in claim 26), Chauvel discloses the system wherein the generated object code comprises a plurality of native instructions, and the runtime manager is further equipped to

- monitor execution of the generated object code for execution requirements of the native instructions (e.g., Fig. 3b – element 56 – For Each Operation,

Divide Energy Consumption by Number of Executions and Store in JVM Profile; [0013] - … based on the number of times operations are executed in the application …; [0031] - … byte-codes that have a wide variant must be separately monitored for wide and normal behavior); and

- update execution requirements of selected ones of the native instructions if said monitoring observes execution requirements for the selected ones of the native instructions to be different from current understandings of the execution requirements of the selected ones of the native instructions (e.g., Fig. 3b – element 56 – For each Operation Divide Energy Consumption by Number of Executions and Store in JVM Profile; element 54 – Increment Associated Register with Energy Consumption; Update Maximum and Minimum Energy Values; [0013] - … an accurate estimation of a performance criteria, such as average time, maximum time, or energy consumption, for the application can be provided …)

28. **As to claim 30** (Original) (incorporating the rejection in claim 26), Chauvel discloses the system wherein the communication interface is a wireless communication interface (e.g., [0010] - … This is particularly true in the case of mobile devices, such as smart phones, personal digital assistants, and the like, which have limited energy and processing resources …)

29. **As to claim 31** (Currently Amended) (incorporating the rejection in claim 11), Chauvel discloses the method wherein determining the initial number of

times to interpretively execute the non-native instructions is based at least in part on a size of the non-native instructions (e.g., [0012] - … by acquiring an application profile that specifies a number of executions of a plurality of operations used in the specified portion of the application … - emphasis added)

30. **As to claim 32** (Currently Amended) (incorporating the rejection in claim 18), please refer to claim **31** above, accordingly.

31. **As to claim 33** (Currently Amended) (incorporating the rejection in claim 26), please refer to claim **31** above, accordingly.

Conclusion

32. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Ben C. Wang
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192